IN THE SPECIFICATION:

Please replace paragraph [0002] with the following amended paragraph:

[0002] In an Integrated Development Environment (IDE), computer users and programmers use a program editor to write computer programs and develop software applications in the form of source code. The source code is conventionally written in a high-level programming language, e.g., C++, the Java® programming language, Pascal, and the like. To run such computer programs in a computer system, a compiler program must convert the source code into executable code or object code.

Please replace paragraph [0011] with the following amended paragraph:

[0011] FIG. 2B depicts a conversion of source code to bytecode in a Java® programming environment;

Please replace paragraph [0033] with the following amended paragraph:

[0033] The IDE software 110 may include a program editor 114, a compiler program 116, and a decompiler program 130. The program editor 114 is a software application that enables a programmer to write and edit computer programs in the form of source code. The source code is written in a high level programming language, e.g., C++, Pascal, the Java® programming language, and the like.

Please replace paragraph [0035] with the following amended paragraph:

[0035] Different types of compiler programs 116 include a traditional "static" compiler, a Java® compiler, or a Just In Time (JIT) compiler. The static or traditional compiler converts source code into executable object code. The Java® compiler converts source code into bytecode. A program known as a "virtual machine" processes the bytecode. The virtual machine comprises an interpreter to execute

instructions in the bytecode and a JIT compiler to compile the bytecode. As such, the bytecode is executed by either the interpreter or compiled by the JIT compiler.

Please replace paragraph [0039] with the following amended paragraph:

[0039] FIG. 2A depicts a conversion of source code 202 using a compiler program 116. The source code 202 comprises one or more programs or files 112 that is generally written in a programming language such as C, C++, Pascal, the Java® programming language, and the like. The compiler 116 is a software program that interprets and converts the source code 202 into object code 204. Such a compiler 116 is known as a static compiler. The object code 204 comprises one or more programs or files used by the operating system 108 or an application program (not shown).

Please replace paragraph [0040] with the following amended paragraph:

[0040] FIG. 2B depicts a conversion of source code 202 in a Java® programming environment. The Java® programming environment uses a Java® compiler 206 to create bytecode 208 from source code 202. The bytecode 208 represents a type of source code 202 that may be processed by a Java® virtual machine program 210 comprising an interpreter 212 and a run time compiler 214, e.g., a Just In Time (JIT) compiler. Specifically, the bytecode 208 is executed by the interpreter 212 or compiled by the run time compiler 214. In contrast to the source code 202, the bytecode 208 is usable in multiple platforms, i.e., operating system 108 and processor 102 combination. The interpreter 212 interprets or maps generalized machine instructions in the bytecode 208 into instructions specific to the processor 102. The run-time compiler 214 compiles the bytecode 208 into executable object code 204 for a specific platform.

Please replace paragraph [0058] with the following amended paragraph:

[0058] FIG. 3G depicts a user interface 302 displaying different levels of optimization of a run time compiler 212. In the Java® programming environment, bytecode 208 is generated from source code 202 using a Java® compiler 206. The virtual machine program 210 then decides whether to interpret or execute the bytecode 208 using the interpreter 212 or compile the bytecode 208 using the run time compiler 214. Suppose a procedure is invoked or called multiple times while executing the bytecode 208. The interpreter 212 would interpret or execute the procedure up to a threshold number of times. After this number has been reached, the run time compiler 214 would then compile the procedure.

Please replace paragraph [0061] with the following amended paragraph:

[0061] FIG. 3H depicts a user interface 320 displaying statistics for inlined procedures contained in source code 202. In the Java programming environment, procedures in the source code 202 may be classified in terms of classes, packages and projects. Such inlining statistics were processed by compilers 116 but not displayed in the prior art. In contrast, the user interface 320 displays the number of times each procedure has been expanded through inlining optimization. Modifications of the current art Java® compiler 206 are needed to provide the results, e.g., number of times a procedure is inlined, from the Java® compiler 206 to the editor 114. Exemplary results are shown for some projects, packages and classes for several procedures.